

BOX PATENT APPLICATION
ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Case Docket No.: PA1084US

EL 470301476US

Sir:
Transmitted herewith for filing is the patent application of
Applicants: **Kent Vuong, Michael Pan, and Ernest Seagraves**
Accelerator Engine for Processing Functions Used in Audio Algorithms

Enclosed are:

- ☐ 22 pages of specification, claims and abstract.
☐ 17 sheets of ☒ informal ☐ formal drawing(s).
☐ A declaration and power of attorney.
☐ An assignment transmittal.
☒ An assignment of the invention to: New Japan Radio Co., Ltd.
☐ Please record the assignment and return to the undersigned.
☐ A certified copy of a _____ application.
☐ An associate power of attorney.
☐ A verified statement to establish small entity status under 37 CFR §§ 1.9 and 1.27.
☐ PTO Form-1449 and copies of cited art.

The filing fee has been calculated as shown below:

For	(Col. 1) No. Filed	(Col. 2) No. Extra	Small Entity		or	Other Than a Small Entity	
			Rate	Fee		Rate	Fee
Basic Fee				\$345.00			\$690.00
Total Claims	18 - 20 = *	0	x \$9 =	\$		x \$18 =	\$0.00
Indep. Claims	3 - 3 = *	0	x \$39 =	\$	or	x \$78 =	\$0.00
Multiple Dependent Claims Present <input type="checkbox"/>			+ \$130 =	\$		+ \$260 =	\$0.00
*If the difference in column 1 is less than zero, enter 0 in column 2			Total	\$	or	Total	\$690.00

☐ Please charge my Deposit Account No. 06-0600 the amount of \$ _____. **A duplicate copy of this sheet is enclosed.**

☒ A check in the amount of \$730.00 to cover the filing fee ☒ and recording of assignment is enclosed.

☒ The Commissioner is hereby authorized to charge payment of the following fees during the pendency of this application or credit any overpayment to Deposit Account No. 06-0600. **A duplicate copy of this sheet is enclosed.**

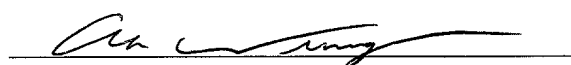
☒ Any additional filing fees required under 37 CFR § 1.16.

☒ Any patent application processing fees under 37 CFR § 1.17.

☐ The issue fee set in 37 CFR § 1.18 at or before mailing of the Notice of Allowance, pursuant to 37 CFR § 1.311(b).

Respectfully submitted,

Dated: April 13, 2000


Aaron Wininger, Reg. No. 45,229
Carr & Ferrell LLP
2225 East Bayshore Road, Suite 200
Palo Alto, California 94303
(650) 812-3400

JC520 U.S. PTO
09/548849
04/13/00

ACCELERATOR ENGINE FOR PROCESSING FUNCTIONS USED IN AUDIO ALGORITHMS

BACKGROUND OF THE INVENTION

1. Technical Field

This invention relates generally to audio digital signal processing (DSP) and more particularly to processing functions used in audio algorithms.

2. Discussion of Background Art

An Inverse Discrete Cosine Transformation (IDCT), which transforms data from the frequency domain to the time domain, requires a pre-multiplication, an inverse Fast Fourier Transformation (IFFT), and a post-multiplication. IDCT is used as one of the last stages of the Dolby®'s third generation audio coding (AC3) decompressing process.

Performing a 128-point IFFT requires 64×7 (=448) radix 2 butterflies, each defined by

$$A' = A - BC \text{ and } B' = A + BC,$$

where A , A' , B , B' and C are complex numbers in the form of $D = d_r + jd_i$. A subscript r denotes the real part, and a subscript i denotes the imaginary part of the complex number.

FIG. 1 shows an audio integrated circuit chip 100 that includes a DSP 102, a Random Access Memory (RAM) 106, a Read Only Memory (ROM) 110, and an Input/Output (I/O) interface 114. DSP 102 provides the main digital signal processing for chip 100 including, for example, filtering and transforming audio samples. RAM 106 is a "memory on chip" used by programs running on DSP 102 to store data relating to input audio samples and the processing performed on those samples. ROM 110 stores

additional data for DSP 102. I/O interface 114 implements various protocols for exchanging, via databus 4005, audio samples with external devices such as analog-to-digital (A/D) and digital-to-analog (D/A) converters, etc.

As audio AC3 and surround sound features are added to chip 100, DSP 102

5 cannot perform as fast as desired, that is, it cannot execute as many million instructions per second (MIPs) as are required to do all of the tasks demanded by chip 100, including, for example, receiving AC3 data, detecting AC3 data error, using IDCT to decode data, and performing audio enhancement and 3D features. One approach to improving DSP
10 102 performance, or to conserving DSP 102 MIPs for other desired functions, accelerates the AC3 decompressing stage. However, this approach requires operations that are tightly coupled with the AC3 algorithm, which in turn requires that a designer be intimately familiar with the AC3 decoding process. Further, the approach is useful for accelerating AC3, but provides no benefits for other audio functions or Moving Picture Expert Group (MPEG) functions.

15 Therefore, what is needed is a mechanism to efficiently improve performance of DSP 102, and thereby performance of chip 100.

SUMMARY OF THE INVENTION

The present invention provides an accelerator engine running in parallel with a DSP in an audio chip to improve its performance. The engine avoids AC3-specific algorithms and focuses on general purpose DSP functions, including biquad filtering and IDCT, that comprise pre-multiplication, IFFT, and post-multiplication. The DSP is therefore free to do other processing while the engine is performing a requested function. The engine utilizes its resources in a pipeline structure for increased efficiency. In the biquad and double precision biquad modes the engine stores data in predefined locations in memory to efficiently access the data. The engine also uses an equation and data values stored in the predefined locations to calculate an audio sample. The calculated result is then stored in a memory location in a way that it can easily be used to calculate the next sample. In a preferred embodiment the engine efficiently saves 15 MIPs in an AC3 based 3D audio product, including 7.5 MIPs in the AC3 decoding process by accelerating the IFFT, and 7.5 MIPs from the 3D processing via a biquad filtering function.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a prior art DSP chip;

FIG. 2 shows a chip utilizing the accelerator engine according to the invention;

FIG. 3 is a block diagram of the accelerator engine;

5 FIG. 4A shows a pipeline structure for the pre-multiplication mode;

FIG. 4B shows a resource utilization map for the pre-multiplication mode;

FIG. 5A shows a pipeline structure for the FFT mode;

FIG. 5B shows a resource utilization map for the FFT mode;

10 FIG. 5C shows C code describing the address generation for both 128-point and 64-point
FFTs;

FIG. 6A shows a pipeline structure for the biquad filtering mode;

FIG. 6B shows a resource utilization map for the biquad filtering mode;

FIG. 6C is a flowchart illustrating how the accelerator engine processes the biquad
filtering;

15 FIG. 6D is a flowchart illustrating how the accelerator engine stores data during the
biquad filtering mode;

FIG. 6E illustrates data in memory during a biquad filtering mode;

FIG. 7A shows a pipeline structure for the double precision biquad filtering mode;

FIG. 7B shows a resource utilization map for the double precision biquad filtering mode;

20 FIG. 7C illustrates data in memory during a double precision biquad filtering mode; and

FIG. 8 is a flowchart illustrating how a chip requests that a function be performed by the
accelerator engine.

DETAIL DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention provides an accelerator engine running in parallel with a DSP for processing functions that are usable by audio algorithms including, for example, AC3, 3D, bass managements, MP3, etc., and that would otherwise be performed by the DSP. Consequently, the DSP is free to process other tasks.

FIG. 2 shows a chip 250 utilizing the invention. Integrated circuit chip 250 is like chip 100 except that chip 250 includes an accelerator engine 200 interfacing via data bus 2003 with DSP 102. Data passing through databus 2003 includes configuration information and data for accelerator engine 200 to perform a requested function.

FIG. 3 is a block diagram of accelerator engine 200 preferably including an RRAM 304, an IRAM 308, a ROM 312, a control register 316, a state machine 320, a multiplier (MPY) 326, a shift/sign extender 328, an ALU 330, and other components. In the preferred embodiment accelerator engine 200 supports the following functions: single biquad filtering; double precision biquad filtering; radix2, 7 passes, 128-point IFFT; radix2, 6 passes, 64-point IFFT, premultiplication 128-word and 64-word configurations, post-multiplication 128-word and 64-word configurations, IDCT 128-word and 64-word configurations, and BFE-RAM read/write.

RRAM 304, depending on the required function, stores different types of values. For example, for a biquad filtering function, RRAM 304 stores filter coefficients. For IFFT and IDCT functions, RRAM 304 stores real parts and IRAM 308 stores imaginary parts of complex numbers required by these functions. Data in RRAM 304 and IRAM 308, where appropriate, contains new values after each time accelerator engine 200 has calculated an audio sample.

ROM 312 is used in IFFT and IDCT modes preferably to store both real and imaginary values for complex numbers required by IFFT and IDCT functions.

State machine 320 generates addresses for RRAM 304, IRAM 308, and ROM 312 and control signals for other circuitry of accelerator engine 200.

MPY 326 is preferably a conventional 24-bit multiplier for multiplying data on lines 3031 and 3035.

Shifter/Sign-Extender 328 shifts its contents to adjust or mask a sign bit on lines 3041 and 3045.

ALU 330 is preferably a conventional ALU which accumulates, adds, or subtracts data on lines 3051 and 3055. ALU 330 can be divided into an ALUA (not shown) and an ALUB (not shown) for use when two ALUs are required, such as in the pre-multiplication, IFFT, and post-multiplication modes.

Multiplexers (MUXes) M01, M03, M05, M07, M09, M11, and M13 perform conventional functions of MUXes, that is, passing selected inputs based on select signals (not shown) provided by state machine 320 and appropriate circuitry.

Latches L01, L03, L05, L07, L08, L09, L11, L13, L15, and L17 perform conventional functions of latches including passing inputs based on clocking signals (not shown) provided by state machine 320 and appropriate circuitry.

This specification uses the following notations in several tables for a pipeline structure and for a resource utilization map. Each column represents a critical resource (RRAM 304, IRAM 308, MPY 326, etc.). Each row represents a phase-one-to-phase-one clock cycle, which lasts 20ns in a preferred embodiment. The tables show what each resource is doing during each cycle. In a resource utilization map, the number in each

entry represents the number of operations a resource is executing during a given cycle. A “0” indicates a resource is idle; a “1” indicates a resource is busy.

In the preferred embodiment the pre- and post-multiplication modes pass data and multiply each data item by a unique complex constant preferably stored in ROM 312, that is,

$$A_n = A_n C_n \quad \text{where}$$

$$A_n = a_m + j a_{in} \quad \text{and}$$

$$C_n = c_m + j c_{in}$$

Parameters a_m and a_{in} represent data in RRAM 304 and IRAM 308, respectively, and C_n are filter coefficients in ROM 312. In the preferred embodiment, n ranges from 0 to 127 (for 128 data points). The pre- and post-multiplication modes are identical except that data is accessed linearly in the pre-multiplication mode and in bit-reverse order in the post-multiplication mode because the IFFT mode leaves its results in bit-reverse order. Coefficients are accessed linearly in both the pre- and the post-multiplication modes.

FIG. 4A shows a pipeline structure for the pre-multiplication mode. In cycle 1 accelerator engine 200 reads b_r from RRAM 304, b_i from IRAM 308, and c_r from ROM 312. In cycle 2 accelerator engine 200 reads c_i from ROM 312. In cycles 3 through 6 MPY 326 performs $b_r * c_r$, $b_i * c_i$, $b_r * c_i$, and $b_i * c_r$, respectively. Accelerator engine 200, instead of using the same c_r and c_i that were read from ROM 312 in cycles 1 and 2, rereads c_i and c_r from ROM 312 in cycles 3 and 4. Rereading these values in cycles 3 and 4 avoids using a register to store the values read in cycles 1 and 2. Further, ROM 312 is available for accessing its data in cycles 3 and 4. ALUA in cycles 6 and 7 performs $A = b_r * c_r + 0$ and $A_0 = A - (b_i * c_i)$, respectively. ALUB in cycles 8 and 9 performs $B = b_r * c_i$

and $B_0 = B + (b_i * c_r)$, respectively. In cycle 10 accelerator engine 200 writes b_r and b_i into RRAM 304 and IRAM 308, respectively. In this pipeline structure, data required for a function in each cycle is made available before the data is needed. For example, b_r and c_r used by MPY 326 in cycle 3 have been made ready to MPY 326 by reading RRAM 304 and IRAM 308 in cycle 1.

FIG. 4B shows a resource utilization map for the pre-multiplication mode, which, starting on any four cycle boundary and taking four cycles, shows the number of operations a resource is executing. For example, MPY 326 and ROM 312 perform four operations (all four 1's), one in each of the four selected cycles 1 to 4. ALUA performs two operations (one in each of cycles 2 and 3) while ALUB performs two operations (one in each of cycles 1 and 4). Data is accessed in cycles 1 and 2 for RRAM 304 and IRAM 308. Therefore, MPY 326 and ROM 312 are utilized 100% of the time (4 operations in four cycles) while each of ALUA, ALUB, RRAM 304, and IRAM 308 is utilized 50% of the time (2 operations in 4 cycles).

Accelerator engine 200 preferably uses seven passes of a radix 2 butterfly to process IFFTs, and employs the following equations:

$$A' = A - BC \text{ and } B' = A + BC,$$

where A , A' , B , B' , and C are complex numbers.

FIG. 5A shows a pipeline structure for the IFFT mode and FIG. 5B shows a resource utilization map for the IFFT mode. The explanations of this IFFT pipeline structure and resource utilization map are similar to the respective explanations of the pre-multiplication mode in FIGs. 4A and 4B. For example, in cycle 1 accelerator engine 200 reads b_r and c_r from RRAM 304 and ROM 312 respectively; MPY 326 is utilized 100% of

the time because in the selected four cycles MPY 326 performs four operations; etc.

Consequently, as shown in FIG. 5B, all resources are utilized 100% of the time.

Accelerator engine 200 performs each pass of the IFFT mode sequentially in both 128-point IFFT and 64-point IFFT. The difference between each pass is the method by which data points are addressed. The addressing schemes for the two modes are similar. FIG. 5C lists C-code describing the address generation for both 128-point FFT and 64-point FFT modes.

In the biquad filtering mode, accelerator engine 200 uses the equation:

$$y_n = b_0x_n + b_1x_{n-1} + b_2x_{n-2} + a_1y_{n-1} + a_2y_{n-2} \quad (1)$$

where the subscript n indicates the current sample number; x_n is the current input sample and y_n is the current output sample; and b_0, b_1, b_2, a_1 , and a_2 are filter coefficients.

Accelerator engine 200 stores filter coefficients in RRAM 304 and input samples and filter states in IRAM 308. In the biquad filtering mode, 48-bit ALU 330 remains as one ALU (instead of being divided into two ALUs: ALUA and ALUB).

FIG. 6A shows a pipeline structure for the biquad filtering mode.

FIG. 6B shows a resource utilization map for the biquad filtering mode. This pipeline structure can be repeated every six cycles and all resources will be used five out of every six cycles.

FIG. 6C is a flowchart illustrating a method for accelerator engine 200 to perform a biquad filtering in accordance with the invention. In step 604, accelerator engine 200 receives, for example, 124 sample data points represented by x_0 to x_{123} . In step 608, accelerator engine 200 stores data in IRAM 308. In step 612, accelerator engine 200 uses equation (1) to calculate y_n for $n=0$ to $n=123$, that is, y_0 to y_{123} , and store them in

appropriate locations in IRAM 308. Accelerator engine 200 then continues to receive, store, and calculate sampled data in respective steps 604, 608, and 612 until all data has been received. Then accelerator engine 200 completes the biquad filtering function in step 620.

FIG. 6D is a flowchart illustrating how accelerator engine 200, in accordance with steps 608 and 612 of FIG. 6C, calculates and stores y_n in IRAM 308 locations for 124 samples of x_n from x_0 to x_{123} . In step 604D accelerator engine 200 stores x_0 to x_{123} in locations 4 to location 127. Accelerator engine 200 also stores values of y_{-2} , y_{-1} , x_{-2} , and x_{-1} in locations 0, 1, 2, and 3, respectively. In this FIG. 6D, locations 0 to 127 are used for illustrative purpose only, any 128 locations, for example, 1 to 128, 2 to 129, or K to $K + 128 - 1$ are applicable. In step 608D accelerator engine 200 uses the values in locations 0, 1, 2, 3, and 4 to calculate y_0 . In step 612D accelerator engine 200 stores the value of y_0 in location 2. Accelerator engine 200 then returns to step 608D to calculate y_1 and store y_1 in location 3, which is one location higher than location 2 storing y_0 . Accelerator engine 200 keeps calculating and storing values of y until accelerator engine 200 is done, that is, accelerator engine 200 calculates and stores values of y_2 to y_{123} in location 4 through location 125, respectively. Accelerator engine 200, in calculating y_0 to y_{123} , uses values of a_2 , a_1 , b_2 , b_1 , and b_0 preferably stored in ROM 312. Those skilled in the art will recognize that calculating y_0 ($n=0$), based on equation (1), requires y_{-2} , y_{-1} , x_{-2} , x_{-1} , and x_0 . The invention uses the zero value for each of y_{-2} , y_{-1} , x_{-2} , and x_{-1} to calculate the first sequence of 124 x_n samples.

FIG. 6E illustrates how IRAM 308 stores a data value for each y_n from y_0 to y_{123} . The "Address" column shows locations from 0 to 127 in IRAM 308. The "Initial Data"

column shows data of y_{-2} , y_{-1} , x_{-2} , x_{-1} , and x_0 to x_{123} in corresponding locations of the “Address” column. Columns $n=0$, $n=1$, $n=2$, $n=3$, . . . to $n=123$, show data in IRAM 308 locations for y_n for $n=0$ to $n=123$, respectively. Box 655 includes values (of y_{-2} , y_{-1} , x_{-2} , x_{-1} , and x_0) that are used to calculate y_0 . Similarly, boxes 659, 663, and 669 include values (y_{-1} , y_0 , x_{-1} , etc.) that are used to calculate y_1 , y_2 , and y_3 , respectively. According to the invention, IRAM 308 locations of values in boxes 655, 659, 663 and 669, etc., are increased by one for each increment of n . For example, box 655 includes values in locations 0 through 4, box 659 includes values in locations 1 through 5, box 663 includes values in locations 2 through 6, and box 669 includes values in locations 3 through 7, etc. Arrow 602 indicates that y_1 is stored in location three, which is one location higher than the location of y_0 . Similarly, arrow 604 indicates that y_2 is stored in location four, one location higher than the location of y_1 . Column $n=123$ shows that y_0 to y_{123} are stored in locations 2 to 125, respectively. Consequently, the invention, while writing the result of y_n (e.g., y_0 in column $n=0$) over the oldest x value (e.g., x_{-2} in the “Initial Data” column) permits the data for calculating y_{n+1} (e.g., y_1) to appear perfectly ordered in the subsequent five locations (e.g., location 1 through location 5). In accordance with the invention, calculating and storing y_n for subsequent sequences of 124 samples of x , that is, calculating and storing y_n for $n=124$ to $n=247$, for $n=248$ to $n=371$, and for $n=372$ to $n=495$, etc., is similar to calculating and storing y_n for $n=0$ to $n=123$. The invention thus uses the same IRAM 308 locations from location 0 to location 127 for calculating and storing y_n for subsequent sequences of 124 samples of x . As discussed above, the invention uses the zero value for y_{-2} , y_{-1} , x_{-2} , and x_{-1} for the first sequence of 124 samples of x . For a second sequence of 124 samples of x the invention uses y_{122} , y_{123} , x_{122} , and x_{123}

for y_{-2} , y_{-1} , x_{-2} , and x_{-1} , respectively. Similarly, for a third sequence of 124 samples of x the invention uses y_{246} , y_{247} , x_{246} , and x_{247} for y_{-2} , y_{-1} , x_{-2} , and x_{-1} , respectively. In calculating y_n , accelerator engine 200 stores filter coefficients preferably in RRAM 304 in order of a_2 , a_1 , b_2 , b_1 , and b_0 .

5 The double precision biquad mode is similar to the (single) biquad mode, but the feedback state is stored and calculated as double precision for greater numerical stability. The equation for the double precision biquad mode is

$$y_n = b_0x_n + b_1x_{n-1} + b_2x_{n-2} + a_1yl_{n-1} + a_2yl_{n-2} + a_1yh_{n-1} + a_2yh_{n-2} \quad (2)$$

10 in which yl_n represents the lower half bits and yh_n represents the upper half bits of y_n . In the preferred embodiment, a double precision term y_n comprises 48 bits, and therefore each term yl_n and yh_n comprises 24 bits.

FIG. 7A shows a pipeline structure for the double precision biquad mode.

15 FIG. 7B shows a resource utilization map for the double precision biquad mode. In this mode accelerator engine 200 operates in a pipeline fashion that repeats every nine cycles.

20 The method for calculating y_n in the double precision mode is similar to that for calculating in the biquad mode except, instead of five values (x_n , x_{n-1} , x_{n-2} , y_{n-1} , and y_{n-2}) accelerator engine 200 uses seven values (x_n , x_{n-1} , x_{n-2} , yl_{n-1} , yl_{n-2} , yh_{n-1} , and yh_{n-2}) as required by equation (2), to calculate each y_n . Further, after calculating y_0 , the invention stores yh_0 and yl_0 in respective locations 2 and 4. Similarly, after calculating y_1 , the invention stores yh_1 and yl_1 in respective locations 3 and 5, each location being one higher than the respective locations of yh_0 and yl_0 . Thus for each y_n , the invention stores yh_n and yl_n each in one location higher than the locations of respective yh_{n-1} and yl_{n-1} .

FIG. 7C shows how accelerator engine 200 stores calculating and calculated values in IRAM 308 for the double precision biquad mode. The “Address” column shows locations from 0 to 127 in IRAM 308. The “Initial Data” column shows data of yh_{-2} , yh_{-1} , yl_{-2} , yl_{-1} , x_{-2} , x_{-1} , and x_0 to x_{121} in corresponding locations of the “Address” column.

Columns 0, 1, 2, 3, ... to 121 show data in IRAM 308 locations for x_n and y_n for $n=0$ to $n=121$, respectively. Box 755 includes values (of yh_{-2} , yh_{-1} , yl_{-2} , yl_{-1} , x_{-2} , x_{-1} , and x_0) that are used to calculate y_0 . Similarly, boxes 759, 763, and 769 include values (yh_{-1} , yh_0 , yl_{-1} , yl_0 , x_{-1} , etc.) that are used to calculate y_1 , y_2 , and y_3 , respectively. According to the invention, IRAM 308 locations of values in boxes 759, 763, 769, etc., are increased by one for each increment of n . For example, box 755 includes values in locations 0 through 6, box 759 includes values in locations 1 through 7, box 763 includes values in locations 2 through 8, and box 769 includes values in locations 3 through 9, etc. Arrow 702 indicates that yh_1 is stored in location 3, which is one location higher than the location of yh_0 . Arrow 703 indicates that yl_1 is stored in location 5, which is one location higher than the location of yl_0 . Similarly, arrows 704 and 705 indicate that yh_2 and yl_2 are each stored in locations one higher than the respective locations of yh_1 and yl_1 , etc. Column $n=121$ shows that yh_0 to yh_{121} are stored in respective locations 2 to 123, while yl_{120} and yl_{121} are stored in respective locations 124 and 125, and x_{120} and x_{121} are stored in respective locations 126 and 127. Consequently, the invention while writing the result of yh_n and yl_n (e.g., yh_0 and yl_0 in column $n=0$) over the oldest values of yl and x (e.g., yl_{-2} and x_{-2} in the “Initial Data” column) permits the data for calculating y_{n+1} (e.g., y_1) to appear perfectly ordered in the subsequent seven locations (e.g., location one through location seven). As in the biquad filtering mode, calculating and storing y_n for subsequent sequences of 122

samples of x , that is, calculating and storing y_n for $n=122$ to $n=243$, for $n=244$ to $n=365$, and for $n=366$ to $n=487$, etc., is similar to calculating and storing y_n for $n=0$ to $n=122$.

The invention thus uses the same IRAM 308 locations from location 0 to location 127 for calculating and storing y_n for subsequent sequences of 122 samples of x . As discussed

above, the invention uses the zero value for y_{h-2} , y_{h-1} , yl_{-2} , yl_{-1} , x_{-2} , and x_{-1} for the first sequence of 122 samples of x . For a second sequence of 122 samples of x the invention uses y_{h120} , y_{h121} , yl_{120} , yl_{121} , x_{120} , and x_{121} for y_{h-2} , y_{h-1} , yl_{-2} , yl_{-1} , x_{-2} , and x_{-1} , respectively.

Similarly, for a third sequence of 122 samples of x the invention uses y_{h242} , y_{h243} , yl_{242} , yl_{243} , x_{242} , and x_{243} for y_{h-2} , y_{h-1} , yl_{-2} , yl_{-1} , x_{-2} , and x_{-1} , respectively. In calculating y_n ,

accelerator engine 200 stores filter coefficients preferably in RRAM 304 in order of a_2 , a_1 , a_2 , a_1 , b_2 , b_1 , b_0 .

FIG. 8 is a flowchart illustrating how chip 100 invokes a function performed by accelerator engine 200. In step 804 chip 100 writes to configuration register 316 to set the required mode and halt accelerator engine 200. In step 808 chip 100 downloads required data from chip 100 to accelerator engine 200. For the pre-multiplication, IFFT, or post-multiplication modes chip 100 downloads data preferably in the order of 0 to 127 and alternating between RRAM 304 and IRAM 308. For the biquad mode, chip 100 downloads coefficients preferably in the order of a_2 , a_1 , b_2 , b_1 , and b_0 in RRAM 304.

Similarly, for the double precision biquad mode, chip 100 downloads coefficients

preferably in the order of a_2 , a_1 , a_2 , a_1 , b_2 , b_1 , and b_0 . Chip 100 also downloads data in the order from 0 to 127, which is the order shown in the "Initial Data" column in FIG. 6E. In step 812 chip 100 determines whether all of the data has been downloaded. If the data is not completely downloaded then chip 100 in step 808 keeps downloading data, but if data

is completely downloaded then chip 100 in step 816 sets the run bit in configuration register 316 so that accelerator engine 200 in step 820 can perform the requested function. Chip 100 in step 824 monitors the status of the done bit in configuration register 316 to determine whether accelerator engine 200 has completed its requested task. In step 828 accelerator engine 200 completes its requested task, and, depending on the mode, chip 100 may or may not set the done bit in configuration register 316. For example, if the requested task is a stand-alone pre-multiplication, then chip 100 sets the done bit, but if the task is an IDCT function then chip 100 does not set the done bit because accelerator engine 200 would continue to perform the IFFT function after completing the pre-multiplication function. In step 832 chip 100, via bus 2003 (FIG. 2), reads data from accelerator engine 200 in linear order except in the IFFT mode where IFFT functions leave data in bit-reverse order.

The invention has been explained above with reference to a preferred embodiment. Other embodiments will be apparent to those skilled in the art after reading this disclosure. Therefore, these and other variations upon the preferred embodiment are intended to be covered by the appended claims.

WHAT IS CLAIMED IS:

1. A method for improving performance of an audio chip including a DSP, comprising the steps of:
 - providing an apparatus having a plurality of elements running in parallel with said DSP;
 - configuring said apparatus to perform a function according to a configuration setup;
 - and
 - employing said apparatus for accessing data from said elements in a pipeline structure to maximize utilization of said elements.
2. The method of claim 1 wherein said function is usable in audio algorithms.
3. The method of claim 1 wherein said function is selected from a group consisting of biquad filtering, double precision biquad filtering, IFFT, IDCT, pre-multiplication, and post-multiplication.
4. The method of claim 1 wherein said plurality of elements includes:
 - a first memory for storing real part data;
 - a second memory for storing imaginary part data;
 - a third memory for storing coefficient data;
 - a multiplier for processing said real part data, said imaginary part data, and said coefficient data; and

an ALU for processing said real part data, said imaginary part data, and said coefficient data.

5. The method of claim 1 wherein in a post-multiplication function, data is accessed in bit-reverse order.

6. The method of claim 1 wherein data is accessed in a four-cycle pipeline structure in a pre-multiplication function, in an IFFT function, and in a post-multiplication function, data is accessed in a six-cycle pipeline structure in a biquad mode, and data is accessed in a nine-cycle pipeline structure in a double precision biquad mode.

7. The method of claim 1 wherein performing a biquad function comprises the steps of:

receiving $N + 1$ samples of data x_n for $n = m$ to $n = m + N$;

storing data including said samples of data in memory locations in a predefined order; and

calculating y_n according to the equation $y_n = b_0x_n + b_1x_{n-1} + b_2x_{n-2} + a_1y_{n-1} + a_2y_{n-2}$.

8. The method of claim 7 wherein said predefined order comprises: y_{m-2} in a location K , y_{m-1} in a location $K + 1$, x_{m-2} in a location $K + 2$, x_{m-1} in a location $K + 3$, and x_m to x_{m+N} in location a $K + 4$ through a location $K + N + 4$.

1 9. The method of claim 8 wherein the step of calculating y_n comprises the steps of:

- 2 (i) using values of y_{m-2} , y_{m-1} , x_{m-2} , x_{m-1} and x_m in respective locations K , $K + 1$, K
3 $+ 2$, $K + 3$, and $K + 4$ to calculate a y_m ;
4 (ii) storing said y_m in said location $K + 2$;
5 (iii) incrementing m by 1;
6 (iv) incrementing K by 1; and
7 (v) returning to step (i).

1 10. The method of claim 1 wherein performing a double precision biquad function
2 comprising the steps of:

- 3 receiving $N + 1$ samples of data x_n for $n = m$ to $n = m + N$;
4 storing data in memory locations in a predefined order; and
5 calculating y_n according to equation $y_n = b_0x_n + b_1x_{n-1} + b_2x_{n-2} + a_1yl_{n-1} + a_2yl_{n-2} +$
6 $a_1yh_{n-1} + a_2yh_{n-2}$.

1 11. The method of claim 10 wherein said predefined order comprises: yh_{m-2} in a location
2 K , yh_{m-1} in a location $K + 1$, yl_{m-2} in a location $K + 2$, yl_{m-1} in a location $K + 3$, x_{m-2} in a
3 location $K + 4$, x_{m-1} in a location $K + 5$, and x_m to x_{m+N} in a location $K + 6$ through a
4 location $K + N + 6$.

12. The method of claim 10 wherein the step of calculating y_n comprises the steps of:

- (i) using values of $y_{h_{m-2}}$, $y_{h_{m-1}}$, $y_{l_{m-2}}$, $y_{l_{m-1}}$, x_{m-2} , x_{m-1} and x_m in respective locations K , $K + 1$, $K + 2$, $K + 3$, $K + 4$, $K + 5$, and $K + 6$ to calculate a y_m ;
- (ii) storing a y_{h_m} and a y_{l_m} of said y_m in said locations $K + 2$ and $K + 4$ respectively;
- (iii) incrementing m by 1;
- (iv) incrementing K by 1; and
- (v) returning to step (i).

13. A method for performing a biquad function comprising the steps of:

- receiving $N + 1$ samples of data x_n for $n = m$ to $n = m + N$;
- storing data in memory locations in a predefined order; and
- calculating y_n according to the equation $y_n = b_0x_n + b_1x_{n-1} + b_2x_{n-2} + a_1y_{n-1} + a_2y_{n-2}$.

14. The method of claim 13 wherein said predefined order comprises: y_{m-2} in a location K , y_{m-1} in a location $K + 1$, x_{m-2} in a location $K + 2$, x_{m-1} in a location $K + 3$, and x_m to x_{m+N} in a location $K + 4$ through a location $K + N + 4$.

1 15. The method of claim 14 wherein the step of calculating y_n comprises the steps of:

2 (i) using values of y_{m-2} , y_{m-1} , x_{m-2} , x_{m-1} and x_m in respective locations K , $K + 1$,

3 $K + 2$, $K + 3$, $K + 4$ to calculate a y_m ;

4 (ii) storing said y_m in said location $K + 2$;

5 (iii) incrementing m by 1;

6 (iv) incrementing K by 1; and

7 (v) returning to step (i).

1 16. A method for performing a double precision biquad function comprising the steps of:

2 receiving $N + 1$ samples of data x_n for $n = m$ to $n = m + N$;

3 storing data including said samples of data in memory locations in a predefined

4 order; and

5 calculating y_n according to the equation $y_n = b_0x_n + b_1x_{n-1} + b_2x_{n-2} + a_1y_{n-1} + a_2y_{n-2}$

6 $+ a_1yh_{n-1} + a_2yh_{n-2}$.

1 17. The method of claim 16 wherein said predefined order comprises: yh_{m-2} in a location

2 K , yh_{m-1} in a location $K + 1$, yl_{m-2} in a location $K + 2$, yl_{m-1} in a location $K + 3$, x_{m-2} in a

3 location $K + 4$, x_{m-1} in a location $K + 5$, and x_m to x_{m+N} in a location $K + 6$ through a

4 location $K + N + 6$.

1 18. The method of claim 17 wherein the step of calculating y_n comprises the steps of:

2 (i) using values of $y_{h_{m-2}}$, $y_{h_{m-1}}$, $y_{l_{m-2}}$, $y_{l_{m-1}}$, x_{m-2} , x_{m-1} and x_m in respective
3 locations K , $K + 1$, $K + 2$, $K + 3$, $K + 4$, $K + 5$, and $K + 6$ to calculate a y_m ;

4 (ii) storing a y_{h_m} and a y_{l_m} of said y_m in said locations $K + 2$ and $K + 4$
5 respectively;

6 (iii) incrementing m by 1;

7 (iv) incrementing K by 1; and

8 (v) returning to step (i).

ACCELERATOR ENGINE FOR PROCESSING FUNCTIONS USED IN AUDIO ALGORITHMS

ABSTRACT OF THE DISCLOSURE

5 An engine for processing functions used in audio algorithms. The engine runs in
parallel with a digital signal processor (DSP) in an audio chip to increase performance for
that chip. Functions performed by the engine include biquad filtering and inverse discrete
cosine transform (IDCT) including pre-multiplication, inverse Fast Fourier transform
(IFFT), and post-multiplication, which would otherwise be performed by the DSP. The
10 DSP is therefore free to perform other functions demanded by the chip. Resources in the
engine are processed in a pipeline structure and are thus highly utilized. Data are stored in
a predefined order to increase efficiency.

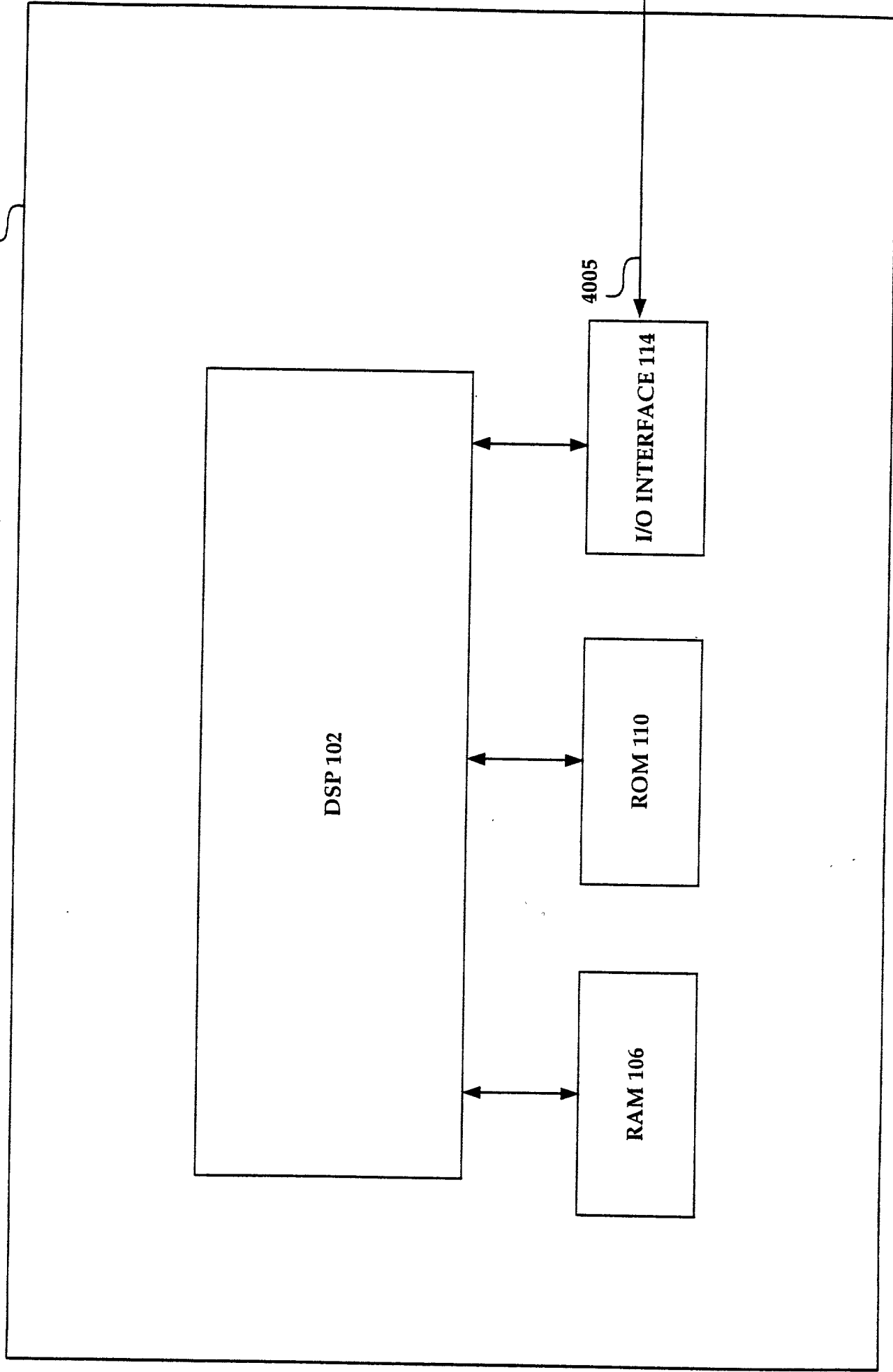


FIG. 1 (PRIOR ART)

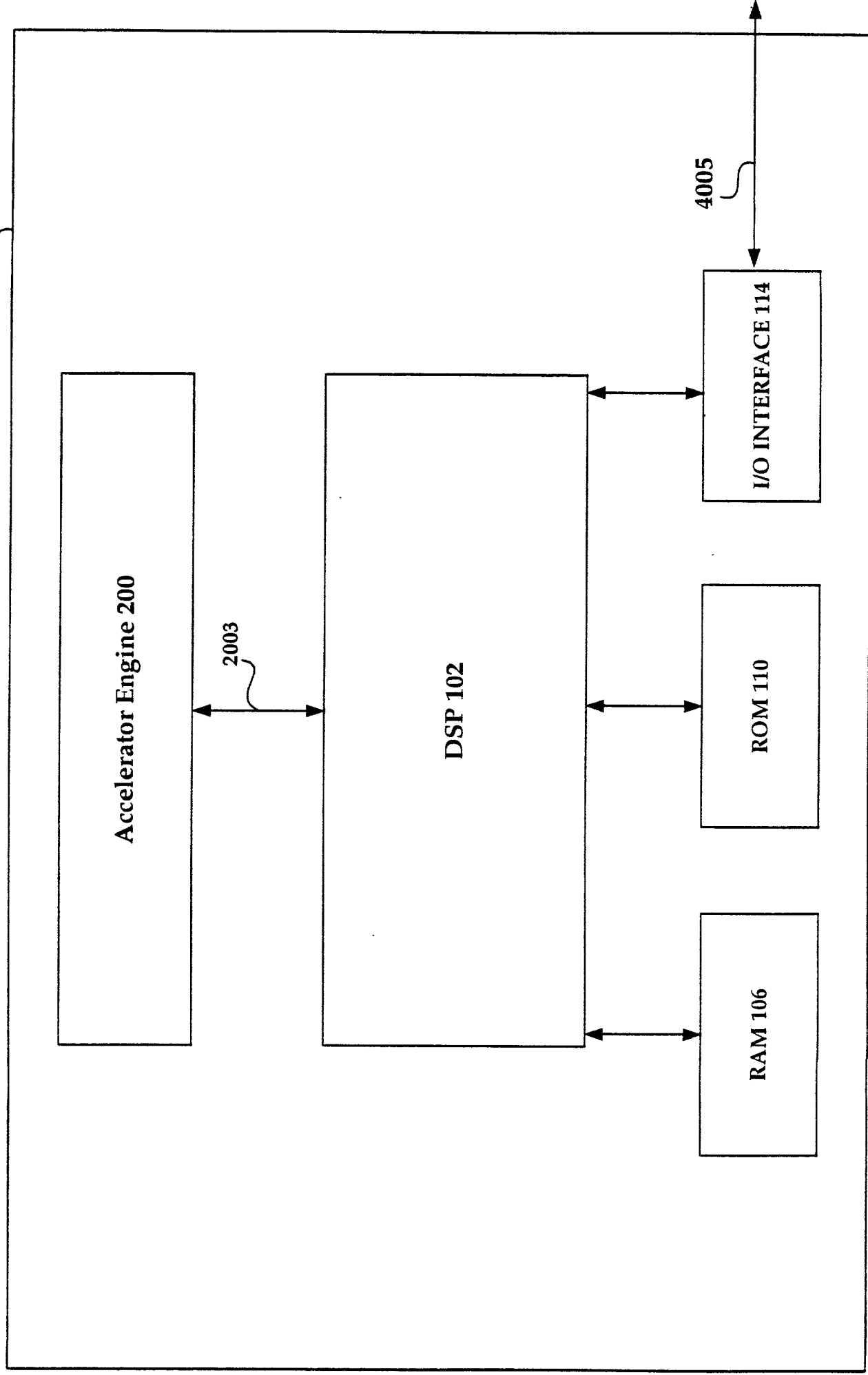


FIG. 2

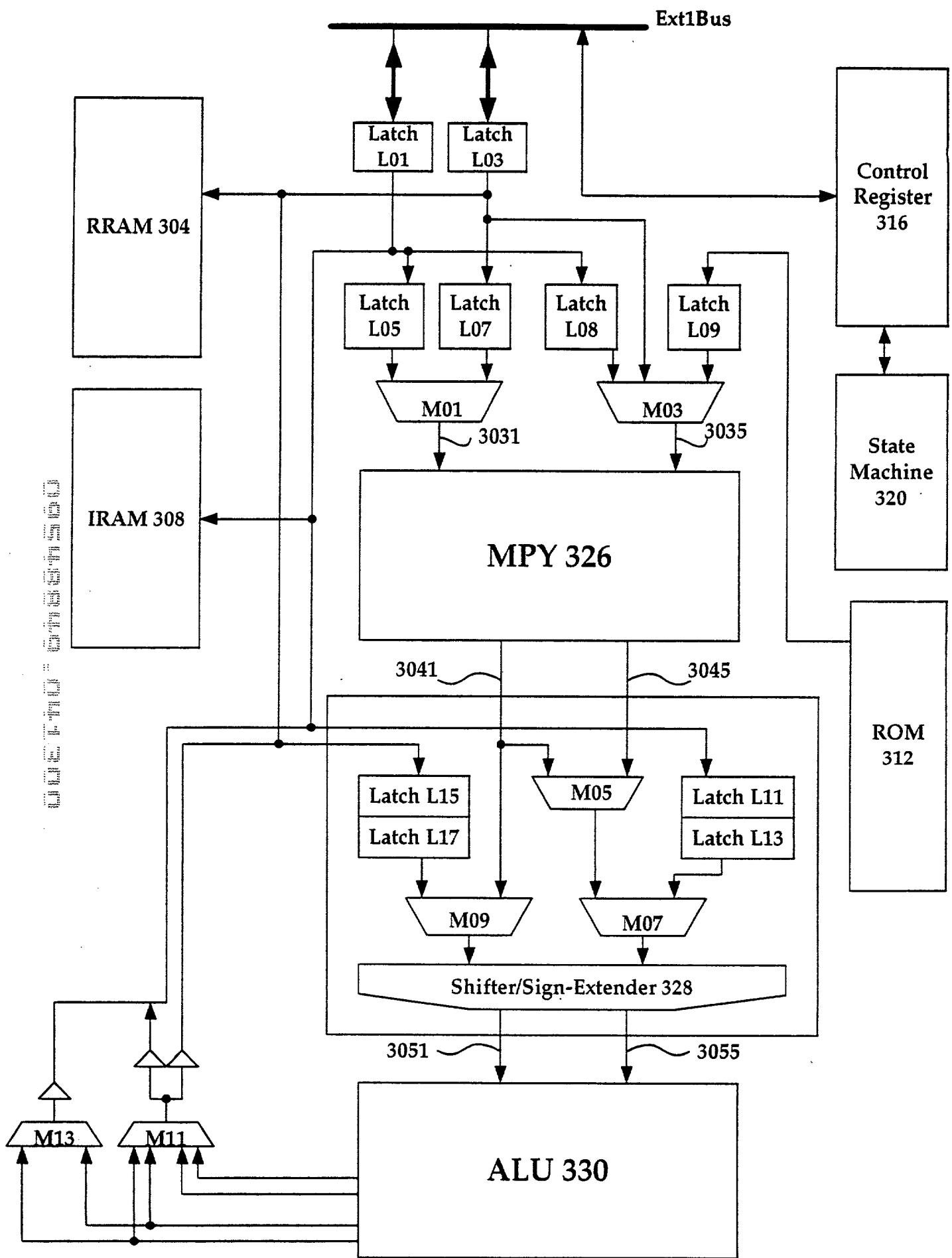


FIG. 3

Cycle	MPY 326	ALU A	ALU B	RRAM 304	IRAM 308	ROM 312
1				read b_r	read b_i	read c_r
2						read c_i
3	$b_r * c_r$					read c_i
4	$b_i * c_i$					read c_r
5	$b_r * c_i$					
6	$b_i * c_r$	$A = b_r * c_r$				
7		$A_0 = A - (b_i * c_i)$				
8			$B = b_r * c_i$			
9			$B_0 = B + (b_i * c_r)$			
10				write b_r	write b_i	
11						
12						

FIG. 4A

Cycle	MPY 326	ALU A	ALU B	RRAM 304	IRAM 308	ROM 312
1	1	0	1	1	1	1
2	1	1	0	1	1	1
3	1	1	0	0	0	1
4	1	0	1	0	0	1

FIG. 4B

Cycle	MPY 326	ALU A	ALU B	RRAM 304	IRAM 308	ROM 312
1				read b_r		read c_r
2					read b_i	read c_i
3	$b_r * c_r$					read c_i
4	$b_i * c_i$			read a_r		read c_r
5	$b_r * c_i$				read a_i	
6	$b_i * c_r$	$A = b_r * c_r$				
7		$A = A - b_i * c_i$				
8		$A_i = a_r - A$	$B = b_r * c_i$			
9		$A_0 = a_r + A$	$B = B + (b_i * c_i)$			
10			$B_i = a_i - B$	write A_i		
11			$B_0 = a_i + B$	write A_0	write B_i	
12					write B_0	

FIG. 5A

Cycle	MPY 326	ALU A	ALU B	RRAM 304	IRAM 308	ROM 312
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1

FIG. 5B

```

Group = 1;
Block = FFT Length / 2;                /* 64 or 32 */
R2P = Log (FFT Length) ;                /* 7 or 6 */
for(i=0;i<R2P;i++)                      /* radix 2 pass counter */
{
    Aiptr=0;                            /* initialize A imaginary pointer */
    Arptr=0;                            /* initialize A real pointer */
    Biptr=Block;                        /* initialize B imaginary pointer */
    Brptr=Block;                        /* initialize B real pointer */
    for(j=0;j<Group;j++)
    {
        for(k=0;k<Block;k++)
        {
            /* perform butterfly here */

            ar = *Arptr;                 /* fetch data */
            ai = *Aiptr;
            br = *Brptr;
            bi = *Biptr;

            rtemp = br * cr - bi * ci;   /* perform complex multiply */
            itemp = br * ci + bi * cr;

            *Arptr++ = ar - rtemp;        /* update and write back data */
            *Aiptr++ = ai - itemp;
            *Brptr++ = ar + rtemp;
            *Biptr++ = ai + itemp;
        }
        Aiptr+=block;                   /* update addresses to next group */
        Arptr+=Block;
        Biptr+=Block;
        Brptr+=Block;
    }
    Block>>=1;                          /* update block size for next radix 2 pass */
    Group<<=1;                           /* update group size for next radix 2 pass */
}

```

FIG. 5C

Cycle	MPY 326	ALU 330	RRAM 304	IRAM 308
1			read b_2	read x_{n-2}
2			read b_1	read x_{n-1}
3	$b_2 * x_{n-2}$		read b_0	read x_n
4	$b_1 * x_{n-1}$		read a_2	read y_{n-2}
5	$b_0 * x_n$		read a_1	
6	$a_2 * y_{n-2}$	$A = b_2 * x_{n-2}$		
7	$a_1 * y_{n-1}$	$A = A + (b_1 * x_{n-1})$		
8		$A = A + (b_0 * x_n)$		
9		$A = A + (a_2 * y_{n-2})$		
10		$A_0 = (A + (a_1 * y_{n-1})) * 2$		
11				
12				write $y_n (A_0)$

FIG. 6A

Cycle	MPY 326	ALU 330	RRAM 304	IRAM 308
1	1	1	1	1
2	0	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	0	1	0
6	1	1	0	1

FIG. 6B

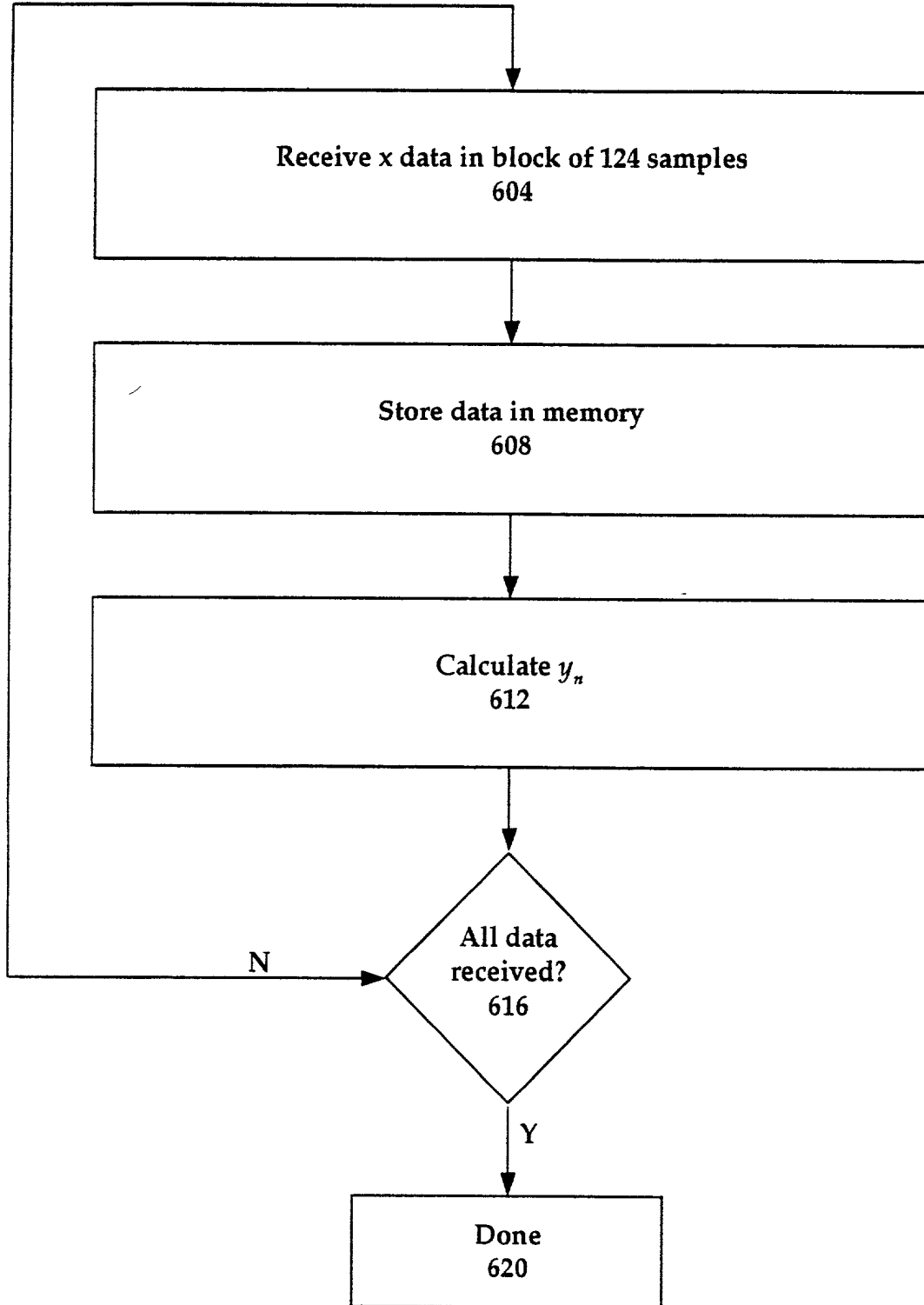


FIG. 6C

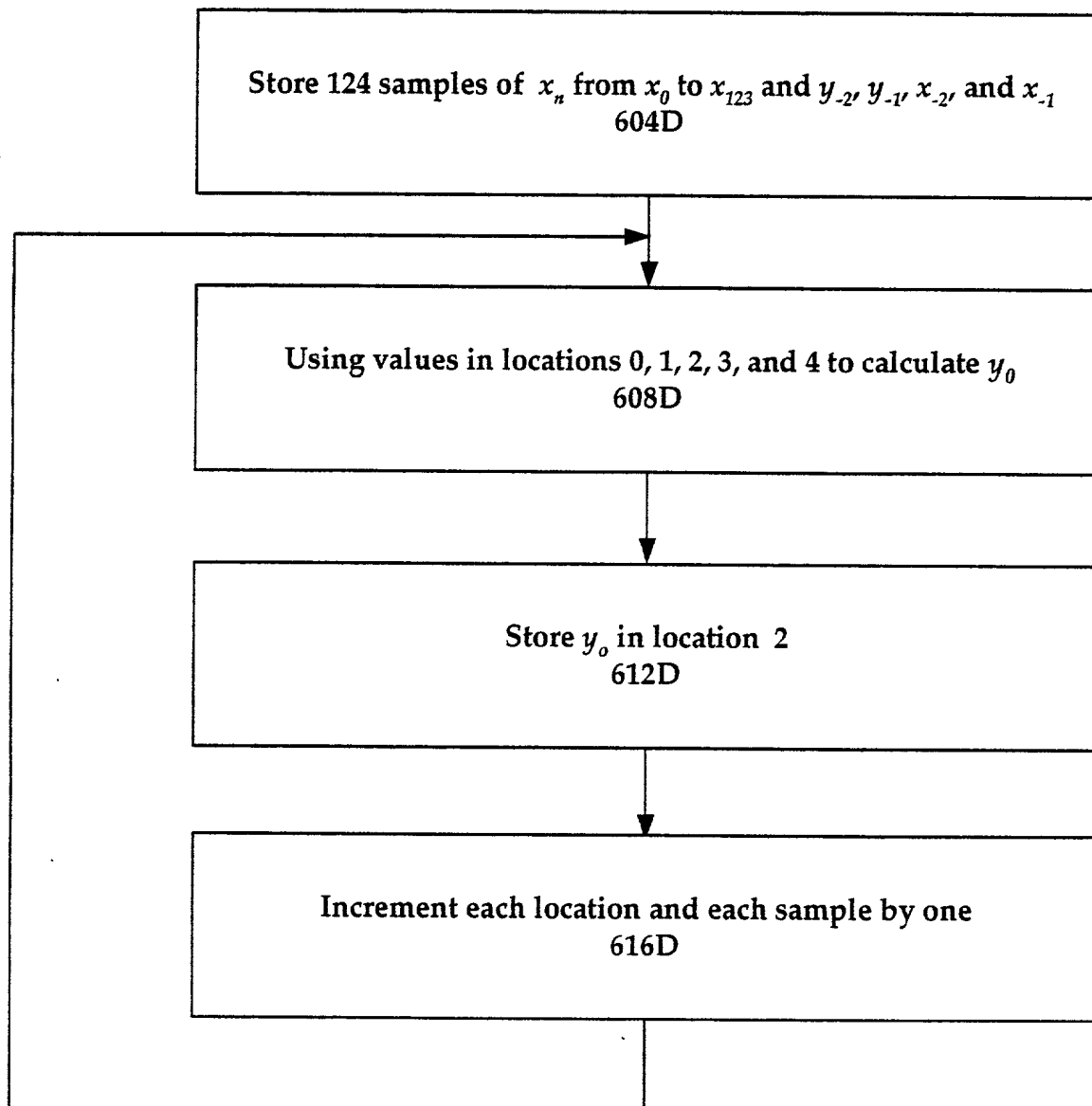


FIG. 6D

Address	Initial Data	Data n=0	Data n=1	Data n=2	Data n=123
0	y ₋₂	y ₋₂	y ₋₂	y ₋₂	y ₋₂
1	y ₋₁	y ₋₁	y ₋₁	y ₋₁	y ₋₁
2	x ₋₂	y ₀	y ₀	y ₀	y ₀
3	x ₋₁	x ₋₁	y ₁	y ₁	y ₁
4	x ₀	x ₀	x ₀	y ₂	y ₂
5	x ₁	x ₁	x ₁	x ₁	y ₃
.	.	.	x ₂	x ₂	y ₄
.	.	.	.	x ₃	.
.
.
.
.
.
.
.
.
.
.
.
124	x ₁₂₀	x ₁₂₀	x ₁₂₀	x ₁₂₀	y ₁₂₂
125	x ₁₂₁	x ₁₂₁	x ₁₂₁	x ₁₂₁	y ₁₂₃
126	x ₁₂₂	x ₁₂₂	x ₁₂₂	x ₁₂₂	x ₁₂₂
127	x ₁₂₃	x ₁₂₃	x ₁₂₃	x ₁₂₃	x ₁₂₃

FIG. 6E

Cycle	MPY 326	ALU 330	RRAM 304	IRAM 308
1			read b_2	read x_{n-2}
2			read b_1	read x_{n-1}
3	$b_2 * x_{n-2}$		read b_0	read x_n
4	$b_1 * x_{n-1}$		read a_2	read yl_{n-2}
5	$b_0 * x_n$			read y_{n-2}
6	$a_2 * yl_{n-2}$	$A = b_2 * x_{n-2}$	read a_1	
7	$a_2 * y_{n-2}$	$A = A + (b_1 * x_{n-1})$		
8	$a_1 * yl_{n-1}$	$A = A + (b_0 * x_n)$		
9	$a_1 * y_{n-1}$	$A = A + (a_2 * yl_{n-2})$		
10		$A = A + (a_2 * y_{n-2})$		
11		$A = A + (a_1 * yl_{n-1})$		
12		$A_0 = A + (a_1 * y_{n-1})$		
13				
14				
15				
16				
17				write $yl_n (B_0)$
16				write $y_n (A_0)$

FIG. 7A

Cycle	MPY 326	ALU 330	RRAM 304	IRAM 308
1	0	1	1	1
2	0	1	1	1
3	1	1	1	1
4	1	0	1	1
5	1	0	0	1
6	1	1	1	0
7	1	1	0	0
8	1	1	0	1
9	1	1	0	1

FIG. 7B

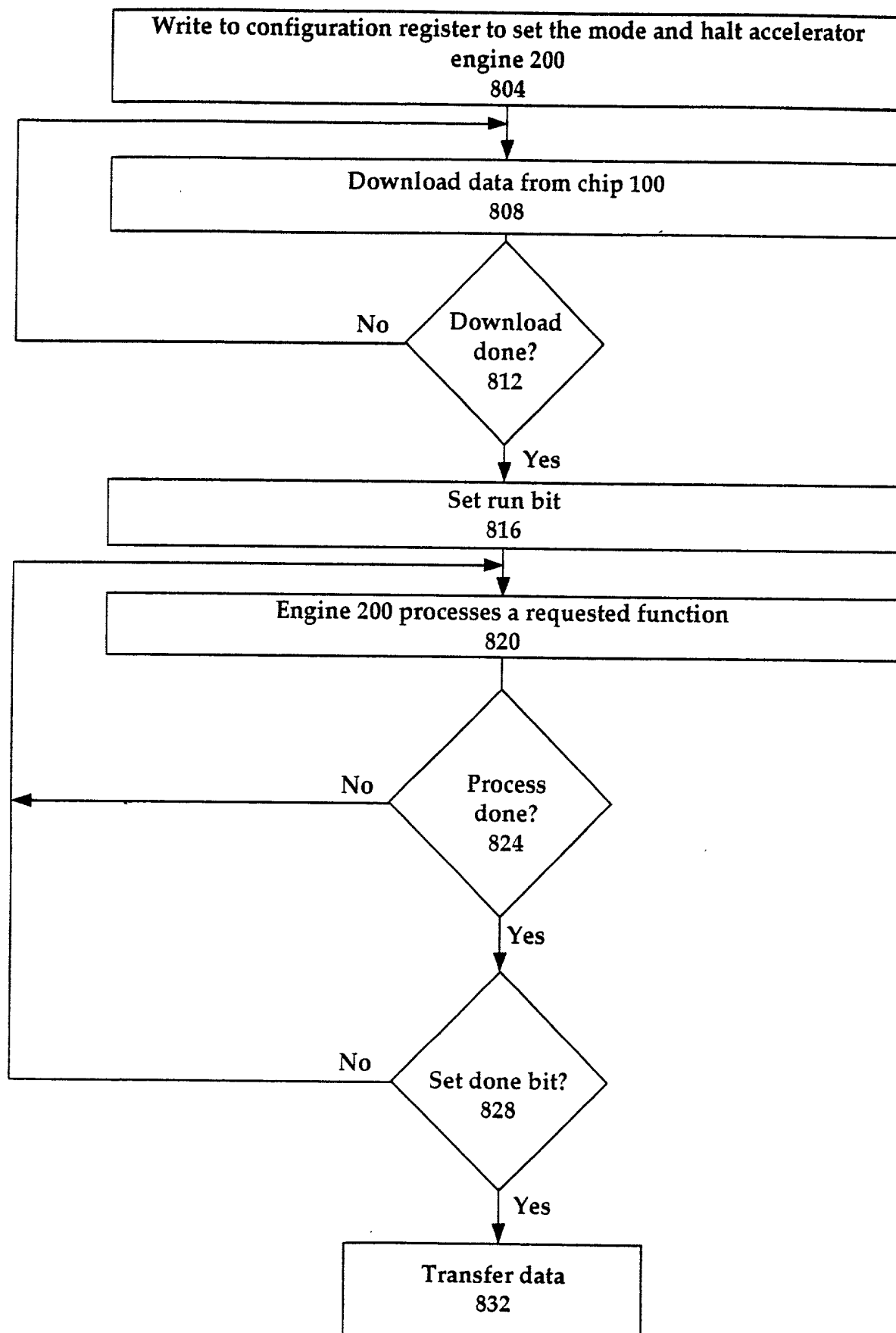


FIG. 8

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am an original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled "**Accelerator Engine for Processing Functions Used in Audio Algorithms**," the specification of which :

☒ is attached hereto.

☐ was filed on _____ as U.S. Application No.
or PCT International Application No. _____
and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment specifically referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code §119(a)-(d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate, or PCT International application, having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s)

Priority Claimed

_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	<input type="checkbox"/> Yes	<input type="checkbox"/> No
_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	<input type="checkbox"/> Yes	<input type="checkbox"/> No

I hereby claim the benefit under Title 35, United States Code §119(e) of any United States provisional application(s) listed below.

(Application Number)

(Filing Date)

(Application Number)

(Filing Date)

I hereby claim the benefit under Title 35, United States Code §120 of any United States application(s), or §365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code §112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application.

(Application Number)

(Filing Date)

(Status -- patented, pending, abandoned)

(Application Number)

(Filing Date)

(Status -- patented, pending, abandoned)

POWER OF ATTORNEY: I hereby appoint the following attorneys and/or agent to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:


John S. Ferrell, Reg. No. 34,593; J. Eppa Hite, Reg. No. 30,266;
Charles B. Katz, Reg. No. 36,564; Gregory J. Koerner, Reg. No. 38,519;
Susan Yee, Reg. No. 41,388; Aaron Wininger, Reg. No. 45,229;
Robert P. Toczycki, Reg. No. 38,341; and
Wendi Schepler, Reg. No. 43,091

SEND ALL CORRESPONDENCE TO:

Eppa Hite
CARR & FERRELL
2225 East Bayshore Road, Suite 200
Palo Alto, CA 94303
TEL: (650) 812-3400
FAX: (650) 812-3444

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: Kent Vuong

Inventor's signature  Dated: Jan 3rd, 2000

Residence 3246 Capriana Circle, San Jose, CA 95135-2302

Post Office Address same as residence Citizenship U.S.A

Full name of second joint inventor, if any: Michael Pan

Second Inventor's signature _____ Dated: _____

Residence _____

Post Office Address same as residence Citizenship _____

Full name of third joint inventor, if any: Ernest Seagraves

Third Inventor's signature _____ Dated: _____

Residence _____

Post Office Address same as residence Citizenship _____

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: Kent Vuong

Inventor's signature _____ Dated: _____

Residence 3246 Capriana Circle, San Jose, CA 95135-2302

Post Office Address same as residence Citizenship _____

Full name of second joint inventor, if any: Michael Pan

Second Inventor's signature Michael Pan Dated: 3/09/2000

Residence 3107 Olive Knoll Place, Escondido, CA, 92027

Post Office Address same as residence Citizenship U.S.

Full name of third joint inventor, if any: Ernest Seagraves

Third Inventor's signature _____ Dated: _____

Residence _____

Post Office Address same as residence Citizenship _____

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: Kent Vuong

Inventor's signature _____ Dated: _____

Residence 3246 Capriana Circle, San Jose, CA 95135-2302

Post Office Address same as residence Citizenship _____

Full name of second joint inventor, if any: Michael Pan

Second Inventor's signature _____ Dated: _____

Residence _____

Post Office Address same as residence Citizenship _____

Full name of third joint inventor, if any: Ernest Seagraves

Third Inventor's signature ET Seagraves Dated: 4-7-00

Residence 900 HIGH SCHOOL WAY, #2105, MT VIEW CA 94041 USA

Post Office Address same as residence Citizenship US